

REMARKS

This communication is in response to the Office Action mailed on June 6, 2003 ("Office Action"), in which Claims 1-18 were rejected. Claims 1 and 13 were rejected under 35 U.S.C. § 102(e) as being fully anticipated by the teachings of Nagel (U.S. Patent No. 6,071,317). Claims 3-4, 7, 9, 15, and 16 were rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of Nagel. Claims 2, 8, 10, and 14 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Nagel as applied to Claims 1, 7, and 13 in view of the teachings of Preisler et al. (U.S. Patent No. 5,675,803). Claims 5-6 and 11-12 were rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of Nagel taken in view of the teachings of Preisler et al. as applied to Claims 2 and 8 in view of the teachings of Boxall et al. (U.S. Patent No. 6,263,456). Claim 17 was rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of Nagel and Preisler et al. as applied to Claim 14 taken further in view of the teachings of Boxall et al. Finally, Claim 18 was rejected under 35 U.S.C. § 103(a) as being unpatentable in view of the teachings of Nagel as applied to Claim 15.

This amendment makes minor changes to the claims directed to eliminating typographical errors and correcting terminology having no clear antecedent basis. Applicants submit that none of the amendments affect the scope of the amended claims. For the reasons hereinafter set forth, applicants respectfully submit that the rejection of Claims 1-18 in view of the teachings of the cited references noted above is in error, should be withdrawn, and this application allowed.

Prior to discussing the reasons why applicants believe that the claims of this application are clearly allowable, a brief discussion of the present invention, followed by a brief discussion of the cited and applied references, is presented. The following discussions of applicants' invention and the cited and applied references are not provided to define the scope or interpretation of any of the claims of this application. Instead these discussions are provided to help the United States Patent and Trademark Office better appreciate important claim distinctions discussed thereafter.

Summary of the Invention

The present invention is directed to providing a system, method, and computer-readable medium for patching **computer application programs** that are not compatible with the **computer operating system** executing the computer application program. More specifically, the system, method, and computer-readable medium determine whether or not the **computer**

application program is compatible with a **computer operating system** executing the computer application program. If the computer application program is determined to be incompatible with the computer operating system, a debugger is started to run the computer application program.

In one exemplary embodiment of the invention, a user starts an application. A determination is made whether the application is compatible or incompatible with the operating system. If compatible with the operating system, the program is run. If incompatible with the operating system, a debugger is started.

Application compatibility with the operating system in one embodiment of the invention is determined by checking application identity information against a database containing a list of applications that are currently known to be incompatible with the operating system. If the computer application program is found in the database, a set of attributes is checked to determine if the particular version of the application is incompatible. If the checked attributes match the ones in the database, the application is determined to be incompatible with the operating system. If the application is not found in the database, the application is determined to be compatible with the operating system. Based on the determination, the application is either run with or without a debugger. Applications that work correctly under the operating system are not required to go through an additional level of execution of the debugger application. Thus, such applications execute faster.

In one version of the invention, if an application is found to be incompatible with the operating system, the operating system starts the debugger application that, in turn, runs the incompatible application. Before loading the incompatible application, the debugger loads a dynamic link library (DLL) that contains patches for the incompatible functions of the application. Specifically, the DLL contains a list of breakpoints specifying the location where the application needs to be patched together with the appropriate code required to patch the application.

In this exemplary embodiment of the invention, when the debugger loads the DLL, a function is called that sets the breakpoints within the incompatible application. A breakpoint may be specified by: (1) the module name and function name; (2) the module name and a memory offset, or (3) an address. The debugger also implements a handler that is capable of modifying the application code that is incompatible. In one embodiment of the invention, the debugger handler is capable of reading and writing memory within the incompatible application. Therefore, the handler is capable of modifying the incompatible code or inserting code into the application to fix the problem code contained within the application. For example, when the debugger reaches a breakpoint within the application, the handler may be called to merely skip a

portion of the incompatible code or the handler may rewrite a portion of the code to contain a certain value in order to make the application compatible with the operating system.

One of the benefits of the use of a debugger application to patch incompatible applications is that this arrangement is very robust. The debugger is capable of monitoring every step an application takes while it is executing. Because the amount of code required to patch an application is generally very small, such as 200 bytes, patches are readily accessible to a user either through a Web site or FTP site.

Summary of the Cited and Applied References

Nagel (U.S. Patent No. 6,071,317)

Nagel is directed to a method and system for modifying computer program logic with respect to a predetermined aspect, namely, whether the logic is compatible with a particular form of data. More specifically, Nagel is directed to a system for determining whether or not computer program logic is Y2K compatible. Alternatively, while not described in any detail, the Nagel method and system can allegedly be used to modify computer program logic so that it is compatible with certain forms of data, such as the currency of a specific nation, indexes that have risen above a prior historical limit, larger or richer data sets than originally anticipated, etc. In summary, Nagel is directed to determining if computer program logic is compatible with a predetermined aspect, namely a specific form of data. **Nagel is not directed to determining if a computer application program is compatible with an operating system.**

Before run time Nagel analyzes the compiled computer program logic of a module for processing involving the predetermined aspect before run time, substantially without decompilation or reference to computer program source code. Before run time, Nagel stores a set of modifications relating to computer logic modifications of the module relating to the predetermined aspect. At run time, based on the stored set of modifications, program control is selectively transferred from the module to a separate logic structure. The modified logical operations are executed with respect to the predetermined aspect. Subsequently, program control is returned to the module. The predetermined aspect may be, for example, a data type, an algorithm type, or interface specification. In the described embodiment of the invention, the predetermined aspect is date-related data--more particularly, logic operations relating to date-related data that is flawed. The Nagel system preferably operates in a mainframe environment wherein the compiled computer program constitutes one or more load modules executing under an operating system. The computer program logic modifications preferably comprise program flow diversions in an original object module. The original object module selectively transfers

logic control to a separate object module to effect modifications to the computer program logic, followed by a return of control to the original object module.

As noted above, and discussed more fully below, Nagel has **nothing whatsoever to do with determining whether or not a computer application program is compatible with a computer operating system executing the computer application program.**

Preisler et al. (U.S. Patent No. 5,675,803)

Preisler et al describes a run-time debugger operation designated a "Fix and Continue" operation. Preisler et al. permits a user to begin a debugging session if an error is encountered in executable code. The user edits the corresponding source code to correct the error and then executes a "Fix and Continue" command without leaving the debugging session. The Fix and Continue command calls a compiler to recompile a source code file with the newly edited text, receives the resulting recompiled object code file from the compiler, uses a dynamic linker to link the recompiled object code into the target application program process, patches the previous versions of the same object code file to refer to the newly compiled code, resets any required variables and registers, and resets the program counter to the line of code being executed when the error was discovered. The debugger then continues the debug session, thereby saving the time it would ordinarily take to quit the debug session, relink and reload the target program, and start the debug session again.

Preisler et al.'s "Fix and Continue" command merely provides for pausing the debugging of a program to allow a programmer to edit source code without leaving the debugger. Preisler et al. nowhere teaches or even remotely suggests adding patches to a program based on the determination that an application is incompatible with an operating system. Nor does Preisler et al. teach or even remotely suggest breakpoint handlers with instructions for patching a program at debugger breakpoints.

Boxall et al. (U.S. Patent No. 6,263,456)

Boxall et al. is directed to a system for remotely debugging client server applications. The system comprises a kicker program in a debugging engine installed on a server and a debugging user interface installed on a client. When a call is made to the application code in a server machine, the kicker program starts the debugging engine to debug the application code. The kicker program stops the debugging engine when the application code has been stepped through or returns. The debugging engine includes means for remotely starting the debugging user interface installed in the client machine.

Like Nagel and Preisler et al., Boxall et al. has nothing whatsoever to do with determining whether or not a computer application program is compatible with a computer operating system executing the computer application program, much less starting a debugger to run the computer application program, if the computer application program is determined to be incompatible with the computer operating system.

The Claims Distinguished

The Office Action has failed to show, and applicants have been unable to find, where any of the cited and applied references teach or even remotely suggest the subject matter of independent Claims 1, 7, and 13, the only independent claims in this application. Claim 1 is a method claim, Claim 7 is a computer-readable medium claim, and Claim 13 is a system claim. More specifically, Claim 1 is directed to a method for patching a computer application program including a plurality of executable steps, Claim 7 is directed to a computer-readable medium having computer-executable instructions for patching a computer application program including a plurality of executable steps, and Claim 13 is directed at a computer system for patching a computer application program wherein the computer system is capable of running an application having a plurality of executable steps. Claims 1, 7, and 13 all recite:

*determining whether or not the **computer application program** is compatible with a **computer operating system** executing the computer application program; and if the computer application program is determined to be incompatible with the computer operating system, starting a debugger to run the computer application program.* (Emphasis added.)

This subject matter is not taught or even remotely suggested by any of the cited and applied references--Nagel, Preisler et al., or Boxall et al. In this regard, paragraph 3 of the Office Action includes the following statement: "Note: analyzing operating system and resources/data thereof (e.g., registry) linking to the program object is equivalent to determining if application is compatible to be executed in such O.S. environment." Applicants respectfully disagree. If applicants understand this statement correctly, it appears to be concluding that Nagel's determining if program logic is data form compliant is a teaching of determining whether or not a computer application program is compatible with an operating system executing the computer application program. Applicants categorically disagree with this conclusion. A determination of whether or not program logic is data form compatible is not the same as determining whether or not a computer application program is compatible with an operating system executing the computer application program. Applicants further submit that a teaching of detecting data form compatibility would not suggest to one of ordinary skill in the art the desirability of determining

if a computer application program is compatible with a computer operating system executing the computer application program and, if incompatible, starting a debugger to run the computer application program. There is simply no teaching or suggestion of the subject matter recited in Claims 1, 7, or 13 of Nagel, much less in Preisler et al. or Boxall et al. Thus, applicants submit that Claims 1, 7, and 13 are clearly allowable.

Since all of the other claims in this application (2-6, 8-12, and 14-18) depend from Claims 1, 7, and 13, respectively, these claims are submitted to be allowable for at least the same reasons that Claims 1, 7, and 13 are allowable. Further, these claims are submitted to be allowable for additional reasons.

Claims 2, 8, and 14 depend from Claims 1, 7, and 13, respectively. Each of these claims recite that the debugger is capable of running the incompatible application by: (a) setting at least one breakpoint within the application indicating a stopping point for the debugger; (b) running the steps of the application through the debugger; (c) monitoring the steps of the application as the application is running through the debugger to determine if the at least one breakpoint has been reached; (d) patching the application when a breakpoint has been reached; and executing steps (b), (c), and (d) until the application has finished running. As noted above, Claims 2, 8, and 14 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Nagel taken in view of the teachings of Preisler et al. Even assuming for purposes of argument that Preisler et al. teaches a "Fix-and-Continue" debugger that meets the recitations of Claims 2, 8, and 14, which applicants categorically deny, this teaching does not make up for the deficiency of Nagel to teach the underlying subject matter recited in Claims 1, 7, and 13. Further, there is no teaching or suggestion in Nagel and Preisler et al., taken alone or in combination, why it would be obvious to combine the individual teachings of these references. More importantly, as noted above, even if these references were combinable, which applicants categorically deny, the resulting combination would not anticipate the subject matter of Claims 2, 8, and 14 when the subject matter of these claims is considered in combination with the subject matter of Claims 1, 7, and 13, the claims from which these claims depend. Consequently, applicants respectfully submit that Claims 2, 8, and 13 are allowable for reasons in addition to the reasons why Claims 1, 7, and 13 are allowable.

Claims 3, 7, and 15 are dependent on Claims 1, 7, and 13, respectively. Each of these claims recites that determining if the application is compatible or incompatible with the operating system comprises: (a) determining if at least one identifying attribute of a plurality of identifying attributes of a computer application program matches at least one identifying attribute of a plurality of identifying attributes of incompatible applications; and (b) if at least one of the

identifying attributes matches, determining that the computer application program is incompatible, otherwise determining that the computer application program is compatible. Since none of the cited and applied references teaches determining if an application is compatible or incompatible with an operating system, clearly, none of the cited and applied references teaches the subject matter of Claims 3, 9, and 15. As a result, applicants respectfully submit these claims are allowable for reasons in addition to the reasons why Claims 1, 7, and 13, the claims from which these claims depend, are allowable.

Claims 4, 10, and 16 are dependent on Claims 3, 9, and 15, respectively. Each of these claims recites that determining if the application is compatible or incompatible with the operating system further comprises storing identifying attributes of incompatible applications and retrieving at least one of the stored identifying attributes for determining if at least one identifying attribute of a plurality of identifying attributes of the computer application program matches at least one of the stored identifying attributes of incompatible applications. Since none of the cited and applied references teaches determining whether or not a computer application program is compatible with a computer operating system executing the computer application program, clearly, none of these references teaches or even remotely suggests the subject matter of Claims 4, 10, and 16. As a result, applicants respectfully submit that these claims are allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

Claims 5, 11, and 17 depend from Claims 2, 8, and 14, respectively. Each of these claims recites that setting the at least one breakpoint within the application comprises: (a) loading a debugger dynamic link library containing a list of breakpoints, each breakpoint having a handler having a set of instructions for patching the application; and (b) the debugger accessing the list of breakpoints from the debugger dynamic link library and setting the breakpoints within the application. Even assuming for purposes of argument that Preisler et al. discloses executing a debugger containing a set or list of breakpoints, each breakpoint having a set of instructions for patching an application, which applicants categorically deny, as recognized on pages 7 and 8 of the Office Action, this teaching does not teach the use of a dynamic link library. Applicants submit that the argument set forth in the Office Action on pages 7 and 8 that a dynamic link library is somehow suggested by the subject matter of Nagel/Preisler et al., taken in combination with Boxall et al., is clearly specious. There is simply no teaching or suggestion in the references of (a) loading a debugger dynamic link library containing a list of breakpoints, each breakpoint having a handler having a set of instructions for patching an application or (b) the debugger accessing the list of breakpoints from the debugger dynamic link library and setting the

breakpoints within an application. Only the application's specification teaches this subject matter. As a result, applicants respectfully submit that Claims 5, 11, and 17 are clearly allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

Claims 6, 12, and 18 depend from Claims 5, 11, and 17, respectively. Each of these claims recite that patching the application when a breakpoint has been reached comprises (a) calling the handler associated with the breakpoint and (b) patching the incompatible application based on the instructions within the handler. Even assuming, for purposes of argument, that taken in the abstract this subject matter is suggested by the references, which applicants categorically deny, clearly, when considered in combination with the claims from which these claims depend, the overall combination is not taught or suggested by the cited and applied references. As a result, applicants respectfully submit that Claims 6, 12, and 18 are allowable for reasons in addition to the reasons why the claims from which these claims depend are allowable.

In view of the foregoing comments, applicants respectfully submit that all of the claims in this application are clearly allowable in view of the cited and applied references. Consequently, early and favorable action allowing these claims and passing this application to issue is respectfully solicited.

Respectfully submitted,

CHRISTENSEN O'CONNOR
JOHNSON KINDNESS^{PLLC}


Gary S. Kindness

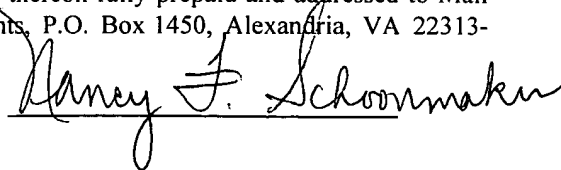
Registration No. 22,178

Direct Dial No. 206.695.1702

I hereby certify that this correspondence is being deposited with the U.S. Postal Service in a sealed envelope as first class mail with postage thereon fully prepaid and addressed to Mail Stop NonFee Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the below date.

Date:

9/5/03


Nancy F. Schoonmaker

GSK:kag/nfs